IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 7, 2020, accepted November 25, 2020, date of publication December 1, 2020,
date of current version December 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041686

The Effects of a Visual Execution Environment and Makey Makey on Primary School Children Learning Introductory Programming Concepts

RAQUEL HIJÓN-NEIRA[1], DIANA PÉREZ-MARIN[1], CELESTE PIZARRO[2], AND CORNELIA CONNOLLY[3]

[1]Computer Science Department, Rey Juan Carlos University, 28933 Madrid, Spain
[2]Applied Mathematics Department, Rey Juan Carlos University, 28933 Madrid, Spain
[3]School of Education, National University of Ireland Galway, H91 TK33 Galway, Ireland

Corresponding author: Raquel Hijón-Neira (raquel.hijon@urjc.es)

This work was supported in part by the Spanish Ministry of Science, Innovation and Universities under Project TIN 2015-66731-C2-1-R, and in part by the Madrid Regional Government through the Project e-Madrid-CM [e-Madrid-CM project is also co-financed by European structural funds [Fondo Social Europeo (FSE) and Fondo Europeo de Desarrollo Regional (FEDER)]] under Grant P2018/TCS-4307.

ABSTRACT The interest of children in learning to program computers has increased dramatically in recent years with the adaptation of new programming languages such as Scratch or game-based approaches. That being so, it is still unclear how best to teach programming concepts to young children. There is a gap in the literature on how to introduce basic programming concepts to children at the primary school level, while taking factors such as the grade level and approach used into account. This paper explores the best approach for introducing basic programming concepts to school children in the 4th, 5th and 6th grades as well as the effects of the approaches on students' learning gains (per concept). The concepts addressed here are those used in a traditional Introduction to Programming course, such as programs, memory and variables, inputs and outputs, conditionals and loops. The paper presents the resulting improvements achieved by the 4th, 5th and 6th graders in a multigroup pretest-posttest design, with a control group (the use of a blackboard as an unplugged approach) and two experimental groups (the use of a visual execution environment (VEE) with a mouse and the use of the VEE with Makey Makey). We present the results exploring the interaction between the grade and approach factors for the 144 children (9-12 years old) enrolled in primary education. The results provide statistically significant data indicating how the children succeeded in learning basic programming concepts according to their grade, the type of approach used, and the programming concept under study.

INDEX TERMS Early years education, primary education, improving classroom teaching, teaching programming.

I. INTRODUCTION

Programming education is seen as a key to the acquisition of skills called "21st century skills", such as creativity, critical thinking, problem solving, communication and collaboration, social/intercultural skills, productivity, leadership and responsibility [1]. Aligned with this fact, interactions with computers are increasing day by day [2]. In a world surrounded by computers and various programs, it becomes necessary to understand the types of problems encountered to be able to produce solutions. Individuals need to know computer languages and have code literacy so that they can participate in daily life activities [3].

Studies in many countries report using Scratch or game-based approaches for teaching programming to young children [4]–[10]. That being so, there are significant difficulties encountered when teaching even basic concepts, such as program constructions [11], loops [12], control structures and algorithms [13]. Difficulties may arise from poor teaching or even a lack of a proper teaching methodology [14], [15]. Teachers need some guidance with regard to efficiently approaching tasks [16], [17].

To address this gap in the literature, the contribution of this paper is aimed at determining approaches for introducing

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

basic programming concepts at the primary school level; this can be achieved by studying factors such as a student's grade, the type of approach used and the interaction between grade and approach. The goal is to understand how these factors affect students' learning gains in general in terms of basic programming concepts. The concepts addressed are those found in a traditional 'Introduction to Programming' course, and these include: programs, memory and variables, inputs and outputs, conditional statements and loops. The types of approach used may be unplugged, on the blackboard, or computerized with a visual execution environment (VEE). A VEE may connect to different tangible interfaces such as Makey Makey connected with fruit versus a mouse interacting with the PC. Therefore, the research questions of this study are as follows:

RQ1. Do children enrolled in 4^{th}, 5^{th} and 6^{th} grades learn basic programming concepts in the same way or are there significant differences?

H1. Children enrolled in higher grades learn more than students enrolled in lower grades.

RQ2. Are children able to learn programming concepts in the same way, or does it depend on how they are taught? For example, would there be differences between children learning without a VEE, those learning with a VEE and a mouse, or those learning programming using Makey Makey?

H2. Students using a VEE with Makey Makey learn more than students using a VEE with a mouse or without technology.

RQ3. Is there an improvement in the test scores for all grades (4th, 5th and 6th grades) and for all types of approaches (control: blackboard, test: PC/mouse, test: PC/Makey Makey)?

H3. Students enrolled in higher grades with a VEE and Makey Makey learn more than students enrolled in lower grades without technology.

RQ4. Are students able to learn some programming concepts better than others depending on their grade and approach?

H4. Some concepts are learned better by students enrolled in lower grades, as they are all novice students without PCs, Makey Makey and previous knowledge, unlike students enrolled in higher grades with more access to technology and previous knowledge.

The paper is organized into six sections: Section II presents related work; Section III focuses on the experiment carried out; Section IV presents the results of the experiment; Section V provides a discussion; and Section VI ends the paper with the main conclusions and lines of future work.

II. RELATED WORK

Much has been written about the most appropriate language and paradigm to use for teaching students about computer programming [18] and the trade-off between choosing a language for its pedagogical suitability or the extent of its use in industry [19]. There has also been an increasing amount of literature on innovative techniques to support introductory programming; suggestions have included the use of robots [20], visual props [21], theatre and even singing [22]. A number of advances have been adopted to help improve students' ability to learn programming, including the use of the ALICE programming environment produced by Carnegie Mellon [23], [24] and many other approaches [25]–[28]. However, considerably less research has been conducted regarding the teaching of basic programming concepts to elementary school children.

This section reviews international approaches for teaching programming at the primary school level, how the approaches are introduced, the use of metaphors for teaching concepts, and program visualization systems for introductory programming education.

A. INTERNATIONAL APPROACHES FOR TEACHING CHILDREN TO PROGRAM

The works of educational theorists such as Forbel, Dewey, Papert, and Montessori argue that learning should be playful, physical, reflective of everyday practices, and self-directed [29]–[31]. Seymour Papert in particular worked to demonstrate that during the activity of computer programming, ''the child is learning how to exercise control over an exceptionally rich and sophisticated 'microworld'' [29]. If a child knows the ''hows'' and ''whys'' behind a concept, they not only have an improved understanding of the information but can develop a solution. The child will create, through their own experimentation and experiences, a connection to the idea that results in them developing a positive outlook on learning [32].

Heintz *et al.* offered a summary of how the USA and several European and Asian countries have started to teach computer programming in their K-12 education systems [33]. It is evident that internationally, computer science (CS) is typically mandatory in primary (elementary) schools and optional in secondary schools (high schools), as depicted in Table 1.

B. WHAT AND HOW TO TEACH PROGRAMMING IN PRIMARY SCHOOLS

A common starting point in many countries and schools is to teach the Scratch computer programming language to children. Developed at the MIT Media Lab [9], [10], Scratch allows students to create interactive programs using programmable instruction blocks [9]. By using Scratch, children learn basic programming features such as conditional statements, events, operators, loops, parallelism, data and sequences [5], [10]. The sequence concept is programmed, for instance, by moving any sprite a short distance. The loop concept, used as a way to repeat instructions several times and represented by the *repeat* command, indicates the number of desired iterations, and the sprite moves accordingly. The conditional concept enables the user to make decisions based on predetermined conditions. The data concept is used for storing, retrieving and updating values. These Scratch approaches focus on visual programming as a means to lower the barrier for small children to learn basic programming

TABLE 1. Interest in teaching computer science [40].

Country	Content	Form	Primary	Secondary
Australia	Digital Technologies	Own subject and integrated	Compulsory	Compulsory
England	Computing	Replace existing subject	Compulsory	
Estonia	Programming	Integrated	Compulsory	Compulsory
Finland	Programming	Integrated	Compulsory	
New Zealand	Programming and Computer Science	Own subject		Elective
Norway	Programming	Own subject		Elective
Sweden	Programming and Digital Competence	Integrated	Compulsory	Elective
South Korea	Informatics	Own subject	Compulsory	Elective
Finland	Computer Science	Own subject	Compulsory	Compulsory
USA	Computer Science	Own subject		Elective
Macedonia	Computers and basics of programing	Own subject	Compulsory	

concepts and to create a fun and motivating environment for students [34]. A previous study introducing very simple concepts to children between 9 and 12 years old using a block-based approach found evidence that there are very little differences in performance between grade levels [35].

A study [36] on tangible computing explained why tangibility matters, especially for your children, because physicality confers cognitive leverage in learning situations. This is relevant, as Scratch can be used with Makey Makey, a simple hardware device for improvising tangible user interfaces also developed at MIT [37]. When using Makey Makey, the user connects fruit, Play-Doh or any other conductive material to Makey Makey, creating a tangible user interface (TUI) that can control the software running on a computer as it receives input from a mouse or keyboard [38]. Integrating a TUI into different teaching methodologies has been proven to increase student engagement due to the interactions between students and the environment [39]. The manipulation of physical objects fosters cognitive learning [40], engagement and active participation to help students clearly comprehend concepts [41]. Baykal *et al.* encouraged designers and researchers to use TUIs with children to obtain many benefits, such as playful learning and embodiment effects [40], [42], [43]. However, other authors [44] have identified situations in which tangible interaction proves less useful and an alternative interaction or hybrid approach provides a more appropriate medium for learning.

Although using Makey Makey is relatively easy, it is not used in many school environments, and minimal research has been conducted on its effectiveness for learning. Lin and Chang used Scratch based on multimedia and Flash with a system based on Makey Makey to boost the motivation of kindergarten kids with cerebral palsy to achieve certain mobility actions [45]. Kafai and Vasudevan studied how middle school students engaged in creating or reusing games in Scratch with Makey Makey and improved their various computational practices through the constructions of their designs [46]. Sun and Han reported in their work that users' enjoyment, interest, excitement, and enthusiasm when using Makey Makey connected with bananas was higher than when they used a standard keyboard input despite a worse performance and lower preference ranking [47]. As a result, García-Peñalvo *et al.* recommended that "even though it may take time to think about how to introduce Makey Makey in the classroom, the benefits will be considerable" [48].

In using techniques such as Makey Makey and visual environments to teach computer programming, theories of education and human activity are further highlighted. Previous studies [49], [50], which were based on Seymour Papert's view of empowering students by mastering programming, concluded that primary school students in the fourth to sixth grades (the former study) and the fifth grade (the latter study) with greater interest in programming perceived it as more meaningful.

The other educational possibilities are developing your own program with Lego WeDo or Mindstorms EV3 robots [7], [51] and unplugged approaches, where students learn programming basics through storytelling or with exercises from Code.org; however, there has not been a proper evaluation of the effects of using these approaches; therefore, the benefits are not yet clear [52].

An approach developed by Brady *et al.* included a variety of technologies, such as physical computing and a rich Internet of things that students can plug into, meant to engage girls in CS, and the approach showed positive effects [53]. Research by Shim *et al.* proposed the use of a robot game environment for learning the fundamental programming concepts of sequences, repeats, conditionals, branches, and parameters, and this technique resulted in a significant increase in programming education efficiency [54]. The systematic review presented by Popat and Starkey also identified the importance of instructional design for developing educational outcomes through coding [55].

C. USE OF METAPHORS TO TEACH CONCEPTS

Metaphors are a crucial factor of thinking [56]. The authors in [57] declared that "the human species thinks with metaphors and learns through stories". As [58] proved in the case of teachers, the key is not only the use of a metaphor but the story provided with the metaphor. Moreover, two theories can be highlighted to support the correct use of metaphors for teaching concepts: conceptual metaphor theory (CMT) and structure mapping theory (SMT).

CMT [59] describes a conceptual metaphor as a cognitive process that occurs when students seek understanding of new

knowledge (the target domain) in terms of a different, already known idea (existing knowledge: source domain) based on their experience with that existing knowledge. Due to its experiential basis, the metaphor can serve as a vehicle for understanding a concept otherwise too abstract for the mind to process [60].

SMT is based on the systematic principle, which claims that the structural view of metaphors is about relations and not because of simple features. Therefore, metaphors can be applied to teach models, stories, concepts and terms [61], [62]. Previous studies have proven the success of using metaphors to teach programming to children [63]. For instance, children may understand a program as a sequence of available instructions that are visualized as an artifact (a turtle drawn on the computer) that moves when the student asks the computer to do so [64]. Children can also understand programming using a cooking metaphor [65]. The ATM metaphor can also be used with the programming concepts of data processing and control structures [66]. The metaphor for a program is a recipe, following the line of programming being expressed as cooking [65]; this metaphor is explained together with scripts in [67] and proven with significant results in [68].

D. PROGRAM VISUALIZATION FOR INTRODUCTORY PROGRAMMING EDUCATION

A review of generic program visualization systems for introductory programming [69] showed a trend of support for engaging modes of user interaction. The results of evaluations largely support the use of program visualization in introductory programming education, but the research to date is insufficient for drawing highly nuanced conclusions with respect to learner engagement, learning tasks accomplished and cognitive load. Specialized systems for helping learners track program execution and providing models of program execution through metaphors or illustrations can be found for CS1 students. For instance, systems for expression evaluation [70], [71] use objects, such as BlueJ [72], Greenfoot [73] recursion [74] and assignment [75]. Specialized systems have the advantage of centering on one topic or a few topics. They may fully explain all difficult aspects, and they can also make use of visual tricks and metaphors that suit the particular topic/s especially well.

III. EXPERIMENT

Previous studies on how to teach programming to elementary school children have shown that there is a lack of agreement with regard to the best approach [76]. This section describes a new experiment to provide some answers for the literature. The experiment is described following the guidelines for reporting software engineering experiments written by [77], [78] with the following subsections: A. Goal, B. Context and participants (experimental units), C. Experimental materials and tasks, D. Variables, E. Hypotheses, F. Experimental design, G. Procedure and tools and H. Validity and reliability.

A. GOAL

The goal is to determine, among unplugged or computerized approaches with a VEE (connected or not connected with Makey Makey), how to introduce basic programming concepts at the primary school level with the highest learning gains. The research questions of this study are as follows:

RQ1. Do children enrolled in 4^{th}, 5^{th} and 6^{th} grades learn basic programming concepts in the same way or are there significant differences?

RQ2. Are children able to learn programming concepts in the same way, or does it depend on how they are taught? For example, are there differences between the children learning without a VEE, those learning with a VEE and a mouse, and those learning programming using Makey Makey?

RQ3. Is there an improvement in the test scores for all grades (4th, 5th and 6th grades) and for all types of approaches (control: blackboard, test: PC/mouse, test: PC/Makey Makey)?

RQ4. Are students able to learn some programming concepts better than others depending on their grade and approach?

B. CONTEXT AND PARTICIPANTS

This study was conducted in a public school in Madrid, Spain. Primary education in Spain spans from the ages of 6 to 12 years. The Ministry of Education establishes core subjects that children must study in every grade. One of them is Castilian Language and Literature, and metaphors are introduced in their fourth year of the curriculum. Therefore, as the methodology to teach programming also relies on metaphors, the earliest year considered for the experiment was the fourth grade (when children are, on average, 9 years of age.)

In Spain, there is a free configuration subject for each autonomous region called ''Technology and Digital Resources to Improve Learning'', which includes programming concepts. Its specific content is set for the primary education curriculum, but each individual school decides on a preferred configuration within the academic program. Consequently, the children participating in this study were those enrolled in the 4^{th}, 5^{th} and 6^{th} grades.

A total of 144 school children at the elementary school level participated in the study. The breakdown was: 46 children (31.9%) from the 4th grade (62.5% male, 37.5% female), 53 children (36.80%) from the 5th grade (40.38% male, 59,62% female), and 45 children (31.25%) from the 6th grade (43,48% male, 56,52% female), with ages ranging from 9 to 12 years.

C. EXPERIMENTAL MATERIALS AND TASKS

Three instructional approaches were used to introduce basic programming concepts to children in this study. The metaphor methodology was common to all the types of approaches. The three approaches adopted were as follows:

- Blackboard and paper exercises, no technology.
- VEE PrimaryCode used with a PC mouse as the interaction method.

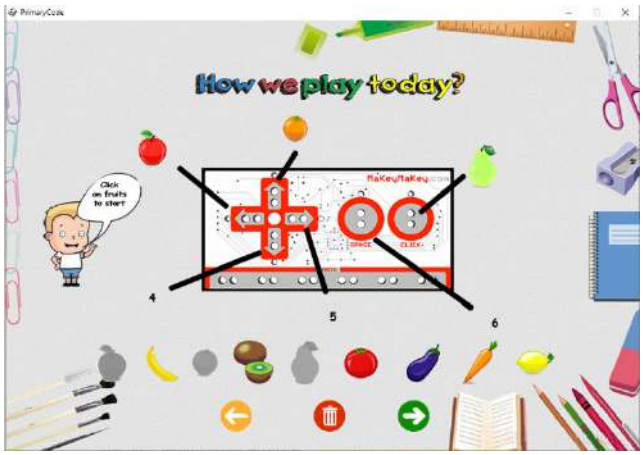

FIGURE 1. PrimaryCode configuration screen when the selected interaction is Makey Makey connected with pieces of fruit (A Play Doh option is also available).

- VEE PrimaryCode used with the TUI of Makey Makey connected with fruit to the PC to allow for interaction with the VEE, where touching the fruit performs the same effects as if the child were touching some keyboard keys (Figure 1).

The difference between types 2 and 3 lies in the interaction method because the VEE PrimaryCode is common to both of them. The second type involves the common use of a PC mouse, and the third type, Makey Makey, can be connected with a plug and play device and configured to use the arrow keys, space button and mouse clicks by touching a connected fruit. For instance, in Figure 1, the left arrow key has been associated with an apple, so when the child uses PrimaryCode and touches the apple, he/she would have pressed the left arrow key, thus making the interaction weird and fun.

In the next subsections, a detailed explanation of the metaphor methodology (1), one application of the VEE PrimaryCode used with a PC mouse (2), another with the VEE PrimaryCode used with Makey Makey connected with pieces of fruit (3), and the tasks carried out by the children regardless of the type of approach (4) are described.

1) METAPHOR METHODOLOGY

In our previous work, a methodology based on the use of metaphors to teach programming in primary education was published [67]. This methodology provides teachers with guidance for teaching the basic concepts of programming to children using the metaphors summarized in Table 2. As indicated in Section II, part C, the key is not only the use of the metaphors in teaching but the scripts by which the metaphor is introduced and how the teachers should use them when trying to explain abstract concepts to students with daily objects [67].

The first block serves to introduce the program, programming, sequence and memory concepts using the metaphor of a recipe, where the memory works as a pantry with boxes and baskets as variables. The second block serves to introduce

TABLE 2. Overview of a smoothie recipe metaphor for use with children.

Block	Concept	Metaphor
1	Program-sequence	Thermomix ™ recipe
	Memory, variables	Pantry, box, barrel
2	Input and output	Mouth and rectum (beginning & end of the digestive system)
3	Conditionals	Intelligent decision making for a fridge
4	Loops	Juicer and/or hand mixer

input and output instructions, with the mouth as input to the digestive system and the rectum as the output. The third block serves to explain conditional instructions as intelligent decision making for a fridges with regard to whether or not there are enough groceries to cook certain meals. The fourth block serves to introduce the loop concept with the simile of a juicer and hand mixer in the kitchen.

This methodology can be used with any resource available to teachers in the classroom. If there is no computer in the classroom, the process will likely take longer because metaphors must be drawn on the blackboard and the students need the scripts and exercises on paper.

If there is one computer in the classroom, the VEE PrimaryCode can be installed and run to support the metaphors. Students can interact with the computer via the mouse. Moreover, if both a computer and Makey Makey [37] are available in the classroom, the metaphors are explained with the VEE PrimaryCode, and students interact with the computer using fruit or Play-Doh as explained previously.

In our previous study on proving the efficacy of the metaphors and the use of MECOPROG [68], 132 primary education students (9–12 years of age) were taught programming by using MECOPROG. At the beginning of the experiment, all students were asked to complete programming concepts and computational thinking tests. During the sessions, all students were taught according to MECOPROG. Finally, they took the tests again. Significant improvements in the results on all the tests were measured, supporting the use of metaphors to teach computer programming concepts to primary education students.

2) METAPHOR METHODOLOGY WITH PRIMARYCODE AND A MOUSE

PrimaryCode is a VEE developed in an ad hoc manner to explain introductory programming concepts to children. This paper presents it for the first time. The VEE is intended to display the metaphor methodology in an interactive way to children. Therefore, to explain the metaphors presented in the previous subsection, it helps to structure the contents similarly in three parts: 1- input and output, 2- conditionals and 3- loops. In addition, the VEE transversally includes the concept of sequence, since it shows the execution of snippets written in pseudocode step by step on the left. Implicitly, the concept of "Program & Programming" is shown, offering several examples where the child can change the inputs and

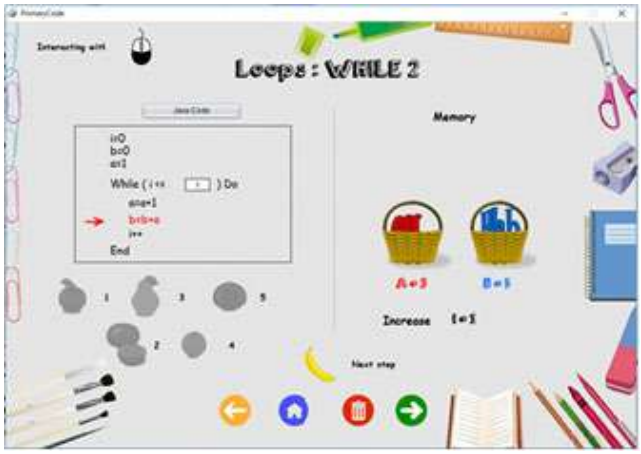

FIGURE 2. PrimaryCode executing a loop statement (left) step by step, where the states of the variables kept on the baskets represent the computer memory (right); in this case, the interaction is done with the mouse (top left).

FIGURE 3. Child executing a snippet on PrimaryCode (left) and watching what happens to the outputs and variable (represented with a barrel), both on the right. In this case, the interaction is done with fruit connected through Makey Makey.

variables values to see what happens in terms of memory or the outputs (screen) as the snippets execute. It offers snippets where students can change the values of variables and inputs on the computer just by selecting them with the mouse. In PrimaryCode, a (sequential, conditional, or input & output) statement is executed step by step on the left-hand by clicking with the mouse, and it runs parallel with the outputs or variables, as shown on the right.

In Figure 2, PrimaryCode shows a loop concept snippet being executed (left) and the baskets representing variables that change (increase or decrease) their contents during every loop iteration (right), and this is all being executed step by step (red arrow pointing out the instruction being executed) by the child clicking the ''next step option'' (bottom) with the mouse. The system offers several types of loops and different examples where children can change the values of variables of the conditional statement and observe what happens. The same is true for the input, output, and conditional statement snippets.

3) METAPHOR METHODOLOGY WITH PRIMARYCODE AND MAKEY MAKEY

The VEE PrimaryCode was designed to introduce children to programming in a fun way; hence, its colorful interface is very child-oriented and presents abstract concepts with simple visual examples such as baskets or barrels for variables or a red arrow for pointing out the next instruction being executed. Thus, the authors were aware of how much children love to use Makey Makey and that using it for learning is definitely valuable [17], [37], [45], [79]. Therefore, the system allows two types of interactions: the first is normal use with the mouse (explained in the previous subsection). In this type of approach, the VEE offers snippets where students can change the values of variables and inputs on the computer just by selecting them by touching the fruit previously connected to it through Makey Makey (Figure 1). In PrimaryCode, a (sequential, conditional, or input & output) statement is executed step by step on the left-hand side by touching the fruit (a banana in this example), and it runs parallel with the outputs and/or variables, as shown on the right (on the right hand side of the screen, a barrel represents the variable that shows the change affected on it); see Figure 3 for an illustration.

PrimaryCode, when used with Makey Makey, offers two options for the tangible interface; it can be configured either with fruit or with Play-Doh. If used with other programs, such as Scratch, any conductive material can be used (aluminum foil, water, etc.).

4) PRIMARYCODE TASKS FOR LEARNING PROGRAMMING CONCEPTS

Table 3 demonstrates the programming concepts inculcated through the exercises performed in PrimaryCode or paper exercises derived from it. There are a total of 31 different exercises that utilize the programming concepts included in Table 2. Some concepts are common to all exercises (sequences, variables), and others include concepts used before (such as inputs, conditions or loops). All exercises have between 2 and 6 different parameters required for execution, and the children can play with different values and observe what happens.

D. VARIABLES

Two different factors were considered in the study as independent variables:

- Grade (three different levels: 4^{th}, 5^{th} and 6^{th}).
- Type of approach (three different levels: unplugged, computerized with a VEE used with Makey Makey, and computerized with a VEE used with a PC mouse)

In addition, interactions between these two factors were considered.

The dependent variables are related to the learning scores of the students obtained on two programming knowledge tests (a pretest at the beginning of the experiment and a posttest at the end of the experiment). These learning scores served

TABLE 3. PrimaryCode tasks for learning programming concepts.

Programming Concept	Includes previous knowledge?	Task in PrimaryCode	Numex
Sequence	Is used across all exercises. The red arrow shows the instruction being executed at every moment.	Students execute little scripts executed via pseudocode step by step and see what happens on the screen and to the variables, memory, and keyboard. They change the inputs using a wide range of values.	31
Memory		Values for memory (barrel, trunk, basket, etc.)appear on all the scripts.	
Output	Sequence Memory	Students select values for variables in the script and see what happens on the PC screen after the outputs are produced.	3
Input	Sequence Memory Output	Students select options for introducing different values into the system and see what happens as the script evolves.	2
Condition	Sequence Memory Output Input	Students select options for the conditions (if-else & switch) and see what happens (on the screen, in memory, asking for inputs, etc.) when the script executes.	8
Loops	Sequence Memory Output Input Condition	Students are offered different scripts with loops (do-while while & for). They execute them interactively with different inputs and see what happens in the variables, on screen, and in the script itself that keeps repeating until a stopping condition is reached!	18

as a quantitative measure for testing the improvements in the students' knowledge of the programming concepts.

E. HYPOTHESES

The hypotheses of the study are as follows:

H1. Students enrolled in higher grades learn more than students enrolled in lower grades.

H2. Students using a VEE with Makey Makey learn more than students using a VEE with a mouse or without technology.

H3. Students enrolled in higher grades with a VEE and Makey Makey learn more than students enrolled in lower grades without technology.

H4. Some concepts are learned better by students enrolled in lower grades, as they are all novice students without PCs, Makey Makey and previous knowledge, unlike students enrolled in higher grades with more access to technology and previous knowledge.

F. EXPERIMENTAL DESIGN

To achieve the goal of the experiment, taking the quantitative nature of the variables and the hypotheses formulated into account was the reason to choose a multigroup pretest-posttest design for the experiment with three groups, one for the control group, known as unplugged, and two test groups, one for each of the approaches under study: computerized with a VEE and Makey Makey and computerized with a VEE and a mouse.

G. PROCEDURE AND TOOLS

The experiment took place in December 2018. The three computer science specialists adopted the metaphor methodology for lesson planning, and the same type of approach was used for each age group (4^{th}, 5^{th} and 6^{th} grade). Each computer science specialist was responsible for one particular type of approach, and each of them taught 4^{th}, 5^{th} and 6^{th} graders.

At the beginning, all students in each of the grades (4^{th}, 5^{th} and 6^{th}) took a pretest concerning basic programming concepts to evaluate their initial knowledge of programming concepts. Students then received 8 sessions (one hour each) on the basics of programming, covering the concepts of programs, programming, sequences, memory, inputs and outputs, conditionals, and loops.

Using a similar approach to that applied by the authors in [54], the children were evaluated on their understanding of 5 programming concepts: sequences, repetitions, conditions and branches, functions and parameters [54]. In this case, a total of 31 visual interactive exercises were carried out by the students, compared to the 10 developed in [54].

Table 4 presents the 12 open-ended questions of the test, indicating the knowledge evaluated by each question. The children answered qualitatively in open text boxes, and they were scored between 0 or 1 for each question according to a rubric. Therefore, the final score of each student was between 0 (minimum) and 12 (maximum). These scores were scaled to a range of 0 (minimum) to 10 (maximum), as that is the usual scoring scale in Spain. The questions were paired in a two-tiered test format in line with the study by Yang [80], with the first type of questions requiring short-answer responses and the second asking for the student's reason for giving the first response or asking for an example.

The time devoted to each session was 60 minutes; however, the time devoted to each particular concept was not predetermined. On the other hand, mastery learning was used [81] given its benefits in the areas of achievement and retention of the content [82]. Therefore, each concept was explained until it could be linked by the children to the previous concepts. As indicated in [67], the teaching methodology advised teachers not to move on to new concepts until previous concepts had been understood.

The teacher recognized when the students understood the concept by performing continuous evaluations. All students had to answer a set of questions and exercises about the concepts during the sessions. It was not just the teacher talking as in a master lecture. Furthermore, irrespective of the approach, all students were given explanations of the concepts (drawn on the blackboard or projected on a screen for the VEE PrimaryCode), and later, the students were asked during the

TABLE 4. Pre-post test questions.

#	Area of knowledge	Question
1	Program, programming, sequence	1. What is a computer program? Can you give an example? 2. What is programming?
2	Output	3. Do you believe that the computer can show information on screen? How? 4. Can you write the approximate instructions for the computer to show "Hello!" on the screen?
2	Input	5. Do you believe that you can type information from the keyboard to the computer? How? 6. Can you write the approximate instruction to save the value "5" typed by a user?
1	Memory	7. Do you believe that the computer can save/store data? What do you think it is for? 8. If the computer has a memory and can save data, do you think that a computer program can modify it? Can you give an example?
3	Conditionals	9. Do you believe that the computer can make decisions? How? 10. If your answer to question 9 was yes, can you write the program instructions for making a decision?
4	Loops	11. Do you believe that the computer can repeat the same task several times? How? 12. If your answer to question 11 was yes, can you write the instructions to repeat an instruction twice?

sessions to complete the same sort of exercises covering the programming concepts being taught (on paper for the control group and on the VEE PrimaryCode for both test groups) and answer questions formulated by the teachers. The differences between the groups were made just to test which type of approach was most adequate for each age group:

- Group 1 (control: blackboard) was the control group; they only used the metaphor methodology explained on the blackboard and performed their exercises on paper (unplugged approach).

- Group 2 (test: PC/mouse) was a test group that used the VEE PrimaryCode with a mouse (described in Section III.C.2). All computers displayed the VEE PrimaryCode on the screen when children arrived at the computer room to prevent any setup time.

- Group 3 (test: PC/Makey Makey) was another test group that used the VEE PrimaryCode with Makey Makey connected to fruit (described in Section III.C.3). As in Group 2, all computers displayed the VEE PrimaryCode on the screen, and the pieces of fruit were already connected to Makey Makey when children arrived at the computer room to prevent any setup time.

At the end of the sessions, all the children took the posttest, which was exactly the same as the pretest.

H. VALIDITY AND RELIABILITY

This paper adapted programming concepts used in earlier research to be appropriate for the target students [52], [83]. Three computer science education specialists, along with three primary teachers (one for each class group involved in the study), together adjusted the questions to the comprehension levels of the targets. Their validity was evaluated by administering them in a test to 144 students (46 fourth graders, 53 fifth graders and 45 sixth graders). The test reliability was high (Cronbach' s $\alpha = 0.81$ for fourth graders, Cronbach' s $\alpha = 0.883$ for fifth graders and Cronbach' s $\alpha = 0.876$ for sixth graders).

All objectives, instructional activities and evaluations were developed by researchers (computer science specialists) and reviewed by the primary education teachers to prevent any possible misunderstanding of the wording of the test following the revision of Bloom's taxonomy [84] (focusing on the lower levels).

Statistical calculations were performed with the IBM SPSS Statistics Version 25 program.

IV. RESULTS

This study focuses on the improvement, first, of the global scores of the individuals within the levels of each factor separately by grade and then by approach type. Second, the improvement is focused on the programming concepts learned, again by grade and by approach type. Subsequently, the study attests to the improvement when these two factors are combined.

A. WITHIN-SUBJECTS EFFECTS

The objective is to know whether the test scores improved as a result of the approach in each grade and of each approach type separately. It is also essential to quantify any possible improvement.

1) RESULTS BY GRADE

To obtain additional information, some statistical parameter calculations are performed for each grade. Table 5 shows the means and standard deviations of the 4^{th}, 5^{th} and 6^{th} grades regardless of the type of approach used for the students (control: blackboard, test: PC/mouse or test: PC/Makey Makey).

Table 6 suggests that there is a significant improvement on the posttest for all grades, whereas their standard deviations show just small increases except for the 6^{th} grade, where it decreases slightly. Figure 4 shows box plots for the three grades on the pre- and posttests. In each box, two values are represented: Q1 or first quartile and Q3 or third quartile. Therefore, each box shows 50% of the cases, and a line inside the box represents the median score. The highest and lowest values for each figure correspond to values that are lower than or equal to Q3+1.5·(Q3-Q1) and greater than or equal to Q1-1.5·(Q3-Q1), respectively.

The Shapiro-Wilk tests used for the three grades conclude normality in all of them. Furthermore, there is no correlation between the samples. In such cases, the t-test for independent samples is used.

Table 6 shows a comparison between the pretest and posttest. There are very significant differences in all the

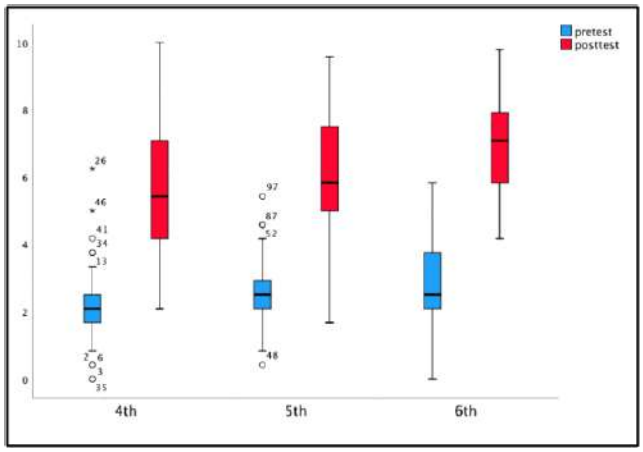

FIGURE 4. Box plots for the students' grades on the pre- and posttests.

TABLE 5. Statistical parameters of the tests for all grades.

	4th grade (n=46)		5th grade (n=69)		6th grade (n=45)	
	M	SD	M	SD	M	SD
Pre	2.11	1.18	2.57	0.95	2.87	1.40
Post	5.61	11.95	6.18	1.87	6.94	1.35

TABLE 6. t-test: Comparison between the pre- and posttests by grade.

	t-test analysis	p-value
4th grade	t=5.826	< 0.001
5th grade	t=6.28	< 0.001
6th grade	t=14.17	< 0.001

TABLE 7. Descriptive statistics for different approach types.

	Control: blackboard (n=46)		PrimaryCode & mouse (n=50)		PrimaryCode & MK MK (n=48)	
	M	SD	M	SD	M	SD
Pre test	2.46	1.26	2.38	1.07	2.74	1.36
Post test	7.34	1.61	5.35	1.49	5.74	1.76

grades. The obvious conclusion is that the participants improved significantly on the test for all approaches.

Some additional information to measure the value of the change produced in all grades is computed using the d-statistics metric proposed by Cohen. All values for the 4th (d = 2.2), 5th (d = 2.46), and 6th (d = 2.99) grades indicate large effects.

2) RESULTS BY APPROACH TYPE

Table 7 shows a descriptive analysis of the means and standard deviations of the test scores for the three types of approaches. The pretest values of the means and standard deviations for different approach types are similar. In contrast, for the posttest, the mean is higher in the control: blackboard group, but its data dispersion is greater than those of the other groups.

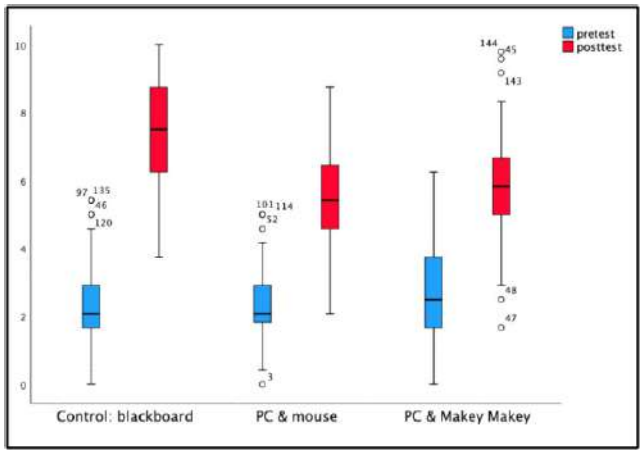

FIGURE 5. Box plots for different groups on the pre- and posttests.

TABLE 8. t-test: Comparison between the pre- and posttests By learning approach.

	t-test analysis	p-value
Control: blackboard (pre-post)	T =5.908	< 0.001
PrimaryCode & mouse (pre-post)	T=6.022	< 0.001
PrimaryCode & MK MK (pre-post)	T=14.175	< 0.001

Figure 5 shows box plots for the three groups on the pre- and posttests Normality can be concluded for the control group ($p > 0.05$ using the Shapiro-Wilk test for the three groups), allowing the use of the t-test for related samples ($p > 0.05$ using the bivariate correlations test).

Table 8 shows significant differences between the pretest and the posttest results with respect to each approach. Consequently, it is deduced that the participants had improved significantly in the test results using all the approach methods.

For further data measuring the magnitudes of the differences between the three learning approaches, the effect sizes in all groups are calculated by the method of Cohen's d statistic [85]. Therefore, the value d = 3.41 for the control group indicates a huge effect. When the metaphor methodology is displayed on the VEE PrimaryCode (interaction with the mouse), the obtained value d = 2.46 corresponds to a huge effect, and the same is true for the value d = 1.93 with Makey Makey. All approaches indicate a large effects, with the highest value reached by the control: blackboard group.

The use of this method for learning the basics of the programming concepts of sequences, memory, outputs and inputs, conditions and loops, regardless of the approach type used, resulted in a significant increase in the efficiency of learning these concepts.

3) RESULTS BY CONCEPT

Figure 6 shows the percentage of students who passed the pre- and posttest by course and by approach. Using Student's t-test for paired samples, significant improvements

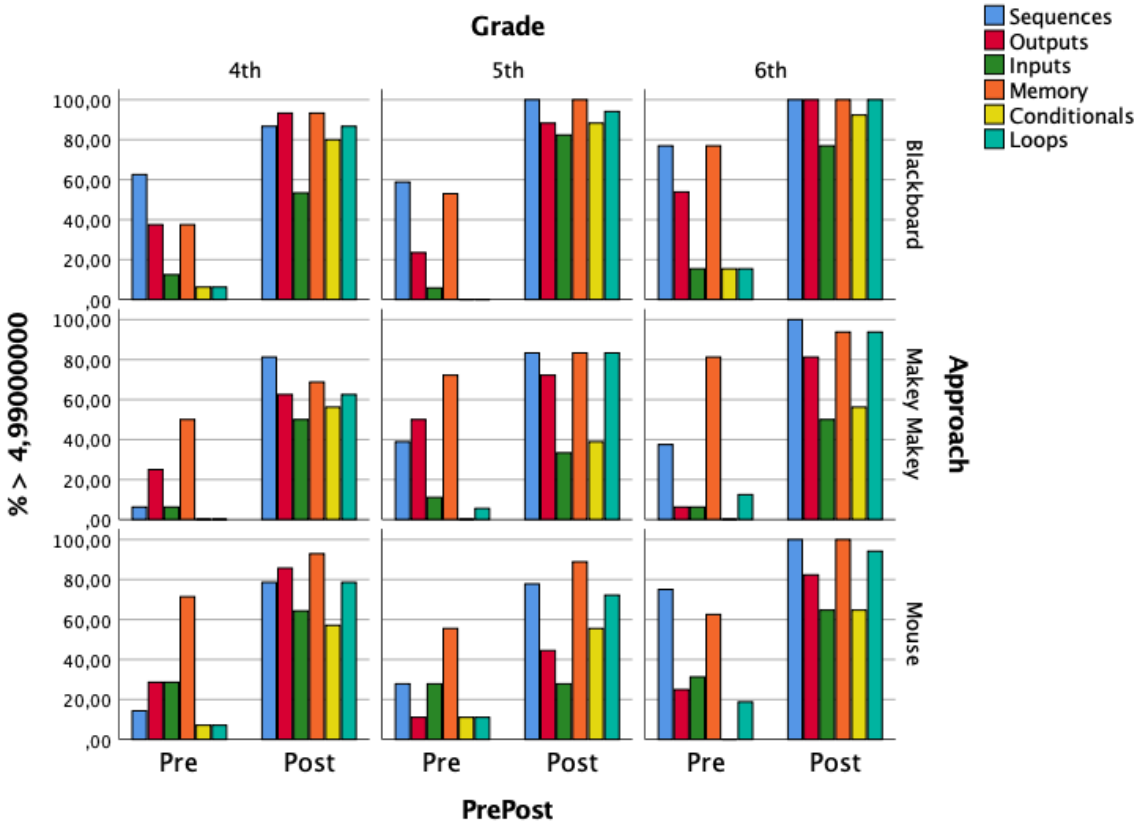

FIGURE 6. Bar chart representing the number of approved students for each approach type in the pre- and posttests, separated by concept.

in the learning of all concepts are observed, both by grade and by approach type.

-4th grade: the concepts with the lowest percentage of passes for the posttest are concentrated in the inputs and conditionals concepts, regardless of the methodology used. The approach type that worked best depends on the concept. Generally, for most concepts, the blackboard worked best, followed by the PrimaryCode used with a mouse and the PrimaryCode used with Makey Makey.

-5th grade: the concepts of sequences, outputs, memory and loops obtained more than a 50% pass rate both for the unplugged approach and for the PC with Makey Makey. For the memory and conditionals concept, ''PC & mouse'' worked better than ''PC & Makey Makey''.

-6th grade: each of the three approach types generally achieved a very large percentage of passes, although the improvement was greatest in the unplugged approach, followed by ''PC & mouse''. In addition, inputs and conditions were the concepts with the lowest number of approvals, although in both cases, they were above 50%. Sequences, memory and loops were the concepts with the highest number of passes for all the approach types, where in most cases 100% of students achieved grades greater than or equal to 5 (out of 10).

Now, instead of analyzing final learning effects by counting the percentage of students who passed, the improvement achieved in relation to the pretest is observed. Thus, the size of the improvement is quantified by Cohen's d statistic. Table 9 shows this sizes of the improvement for all the grades together (4th, 5th and 6th) and divided by the approach type, thus for ''Blackboard'', ''PC & Makey Makey'' and ''PC & mouse'', it can be said that:

-There are at least very large effects for concepts of loops, conditionals, sequences and outputs, regardless of the approach type used.

- The memory concept only has a very large effect when using the ''Blackboard'' and ''PC & mouse'' approaches.

- The inputs concept only has a very large effect when using ''Blackboard''.

Table 10 shows the sizes of the improvements for all the approach types together (''Blackboard'', ''PC & Makey Makey'' and ''PC & mouse''), split by grade; thus, for the 4th, 5th and 6th graders, it can be said that:

-There are great improvements for all the grades for the concepts of (in descending order) loops, conditionals, sequences and memory.

- The output concept has very large effects only for the 6th and 4th graders.

TABLE 9. Cohen's d-value for different concepts by interaction type.

	Blackboard	PC & MK MK	PC & Mouse
Sequences	1,39	1,61	1,37
Outputs	1,74	1,15	1,12
Inputs	1,59	0,92	0,6
Memory	1,96	0,8	1,12
Loops	2,95	1,74	1,61
Conditionals	2,42	2,17	1,78

TABLE 10. Cohen's d-value for different concepts by grade.

	4^{th}	5^{th}	6^{th}
Sequences	1,38	1,46	1,58
Outputs	1,32	1,01	1,64
Inputs	0,86	0,84	1,31
Memory	1,17	1,33	1,42
Loops	1,81	1,85	2,49
Conditionals	1,74	1,87	2,04

TABLE 11. Statistical parameters of the diftest variable for grade and approach type.

	d.f.	F	Sig	Partial eta squared	Observed Power
Grade	2	1.888	0.155	0.027	0.387
Approach	2	15.716	< 0.001	0.189	0.999
Grade*Approach	4	2.935	0.026	0.078	0.763
Error	135				

- The inputs concept, the hardest of all, only achieves very large improvements in 6^{th} grade.

B. EFFECTS BETWEEN SUBJECTS

Next, the objective is to determine if the grade that the students are in, the approach type used and/or a combination of both influence the magnitude of improvement of the test scores, assuming that there is one.

The profile plot shown in Figure 7 is a graphical output from Table 11. It shows how the approach used and grade influence the mean of the diftest variable. It seems that there is some kind of interaction, as the lines are not parallel. Furthermore, the line for the means corresponding to 5^{th} grade shows more abrupt changes than the other lines due to the approach used. A two-way ANOVA calculation can be used to examined the effects of the approach and grade level on the test scores:

$$y_{ij} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \varepsilon_{ij}$$

where y_{ij} is the dependent variable, computed as the difference between the post- and pretest scores (named diftest); μ corresponds to the model mean; and α_i is the main effect of level i of the grade (i = 1, 2, 3). β_j is the main effect of level j of the approach (j = 1,2,3). $(\alpha\beta)_{ij}$ is the interaction effect of treatment (i,j) from both factors, grade and approach. ε_{ij} is the random error.

First, the normality and homogeneity (the population variances are equal across all subpopulations) of the dependent variable are satisfactorily checked. These assumptions are necessary to ensure the effectiveness of the results.

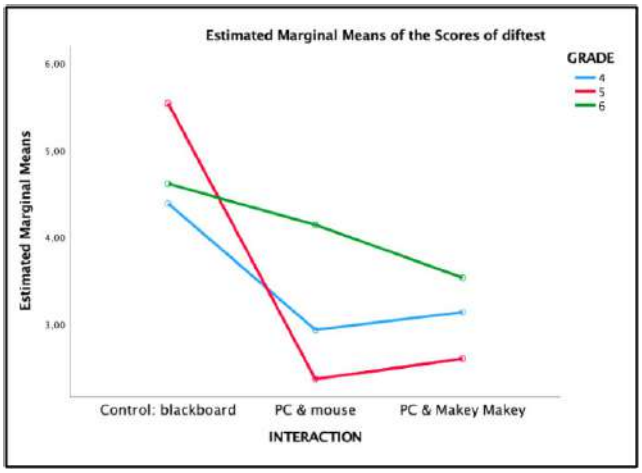

FIGURE 7. Estimated marginal means of the *diftest* variable for each grade and approach type.

TABLE 12. Two-way ANOVA with interaction.

	Control: Blackboard		PC & Mouse		PC & MK MK	
	M	SD	M	SD	M	SD
4^{th}	4.38	1.91	2.93	1.54	3.13	1.89
5^{th}	5.53	1.67	2.37	1.07	2.60	1.53
6^{th}	4.61	1.23	4.14	2.11	3.53	2.14

Table 12 shows that there was a statistically significant interaction between the effects of the approach type and grade level on the scores ($p = 0.026$). The main effects analysis in two-way ANOVA showed that the approach does indeed influence the test score ($p < 0.001$) but also that there are no differences between grades ($p = 0.155$).

Table 13 presents the results from deeper studies of the simple effects, and these show the parameters used in one-way ANOVA, split by grade. The factor is the approach used.

TABLE 13. One-way ANOVA by grade. Factor: Approach/Methodology.

Grade	d.f.	F	Sig.
4^{th}	2	3.063	0.057
5^{th}	2	25.742	< 0.001
6^{th}	2	1.160	0.323

The effect of the approach used is statistically significant in 5^{th} grade ($p < 0.001$). This means that the approach used influences the results in some way. Therefore, a post hoc analysis for 5th grade yielded more information. Tukey's HSD test, which compares each approach with every other approach twice, was used.

Table 14 shows that the comparisons between "control: blackboard" and "test: PC & mouse" and between "control: blackboard" with "test: PC & Makey Makey: are statistically significant ($p < 0.001$).

1) RESULTS BY CONCEPT

Table 15 shows an informative result for a two-factor ANOVA with interaction. An "X" in a given box denotes a p-value larger than 0.05, implying that there are no differences among

TABLE 14. Post hoc comparison: Tukey's HSD.

	Mean Difference	Std. Error	Sig.
Control: blackboard vs. PC & mouse	3.168	0.489	<0.001
Control: blackboard vs. PC &MK MK	2.933	0.489	<0.001
PC & MK MK vs. PC & mouse	0.235	0.482	0.878

TABLE 15. Summary table for a two-way ANOVA with interaction (by concept).

	Grade	Approach	Grade*Approach
DifSequence	X	X	X
DifOutput	X	X	✓
DifInput	✓ (*)	✓(**)	✓
DifMemory	X	✓(**)	X
DifCondition	X	✓(**)	X
DifLoops	X	✓(**)	✓

(*) Between 5° and 6° courses.
(**) Differences are found in "Blackboard vs MK MK" and/or "Blackboard vs Mouse".

the different levels of the factor studied. A check symbol in the box denotes a p-value lower than 0.01, which implies significant differences have been found among the levels of the factor.

A deeper study is necessary when a significant interaction is found, and future research would be to analyze the Grade factor, which is currently fixed and studied separately. A one-factor ANOVA is carried out, and the results show the following:

- In 4th grade, the approach type is not very important for learning the concepts of sequences, outputs and conditionals, but it is more effective to use ''Blackboard'' for memory. It is also more effective to use ''Blackboard'' and ''PC & mouse'' for loops.

-In 5th grade, all approach types work well for sequences, but for the rest of the concepts, it is most effective to use ''Blackboard''.

-In 6th grade, all approaches work well for all the concepts except for loops, where the use of ''Blackboard'' is most effective.

V. DISCUSSION

In recent decades, there has been increasing interest in how to teach programming to children. Several approaches have been tried, as reviewed in the literature. In this experiment, three approaches were under study: an unplugged approach using only the blackboard and a computerized approach using a VEE with or without Makey Makey. A multigroup pretest-posttest design has been followed to answer the research questions described in subsection A along with their practical implications, and the threats to the validity of the experiment are explored in subsection B.

A. ANSWERS TO RESEARCH QUESTIONS

RQ1. Do children enrolled in 4th , 5th and 6th grades learn basic programming concepts in the same way or are there significant differences?The results for the grade factor are provided in Section IV.A.1. It has been recorded how the participants improved significantly on the test for all grades with large effects. It seems that there are no significant differences between the grades. This invalidates H1, as students enrolled in higher grades do not learn more than students enrolled in lower grades, which could have been considered the expected outcome. This has an important practical implication, as teachers should not presuppose that students in lower courses are not going to be able to understand the concepts; they could try, as it is possible that they may be able to understand them as well as students in higher grades.

RQ2. Are students taught without a VEE or with a VEE with a mouse or Makey Makey able to learn basic programming concepts in the same way? Are there significant differences?The results for the approach factor are provided in Section IV.A.2. All approach types (with a VEE with a mouse, with Makey Makey, or just the blackboard) have large effects on the students' learning. The highest improvement is registered in the control: blackboard group. This result invalidates H2, as students using a VEE with Makey Makey do not necessarily learn more than students using a VEE with a mouse or without technology. These results could be surprising at first or even contradictory, but they support other studies proving the benefits of unplugged approaches for teaching programming [86], [87]. As a practical implication, it is highlighted that even in situations in which technology cannot be used, it is possible to effectively teach programming to children.

RQ3. Are the improvements in the test scores the same for all grades (4th , 5th and 6th grades) and for all types of approaches (control: blackboard, test: PC and mouse, test: PC and Makey Makey)? The results for the interaction between the grade and type of approach factors are provided in Section IV.B. The approach used influences the results in some way. This indicates that there is an interaction between the grade and approach factors, where only in 5th grade is the control: blackboard approach most effective. This implies that one cannot expect that students enrolled in higher grades with a VEE and Makey Makey will learn more than students enrolled in lower grades without technology, thus invalidating H3. As a practical implication, teachers of 5th grade students who can choose between different resources for teaching programming to the children are advised to use an unplugged approach to achieve highest learning gains.

RQ4. Are students able to learn some concepts better than others depending on their grade and the type of approach used? The results for the interaction between the grade and approach factors by concept are provided in Sections IV.A.3 and IV.B.1. H4 is supported by the results, as it can be distinguished that concepts such as loops, conditionals and sequences show very large effects for all the

approach types and grades. While the memory concept only yields a very large effect using the blackboard or "PC & Mouse" regardless of the grade level, the inputs concept only yields a very large effect using the blackboard and for the 6th grade, and the outputs concept only yields a very large effect for the 6th and 4th grades regardless of the approach type used. This has practical implications for teachers of students enrolled in the 4th, 5th, and 6th grades who want to teach those programming concepts, as they can provide additional help for some concepts and use the most adequate approach for each case. For instance, to teach the memory concept, it is recommended to use the unplugged approach or the VEE with a mouse.

B. THREATS TO VALIDITY

There are some threats to the validity of this study [88], [89]:

- **Construction validity**: The use of a computer can present some novel effects for students, since many of them abandon their routine and find something else that is interesting and attractive to them. This innovation can create enthusiasm that makes it easier a teacher to achieve his/her goals [90].
- **External validity**: Given that it is difficult to have two different treatments in the same class, the split of the students, albeit random, was limited to keeping the students of the same group together in their class level while using the same approach (unplugged, a VEE with Makey Makey or a VEE with a mouse).
- **Internal validity**: Since this study is a true experimental design, specifically a multigroup pretest-posttest design including a control group [91], the degree of control over the experiment is great. However, sometimes, with human participants, it is not possible to have a very high degree of control [92]. Furthermore, a historical threat may be that, in the entire experiment, the same teachers remained with the same group of students, which may influence the grades obtained on the tests.
- **Conclusion validity:** Statistical conclusion validity "is concerned with sources of random error and with the appropriate use of statistics and statistical tests" [88], so there is no perfect validity. At the moment in which an inferential study of the data is carried out, type 1 errors (rejecting the null hypothesis incorrectly) and type 2 errors (accepting a null hypothesis that is false) may be present, although the study does try to minimize them.

VI. CONCLUSION

This paper contributes to the CS education literature with an original study relating the grade in which children are enrolled, 4th, 5th or 6th grade, the type of educational approach used, and the students' gain in knowledge of basic programming concepts when being taught in primary education. The three types of educational approaches were as follows: Group 1- control, no technology, reproducing what was done with a VEE PrimaryCode using unplugged approaches with a blackboard and paper exercises; Group 2- where students interacted with a PrimaryCode program using a mouse; and Group 3- students interacted with a PrimaryCode program using Makey Makey.

All types of approaches provided satisfactory results with regard to student learning, and there were no statistically significant differences among the three types of approaches. The data confirmed that even when the students were prone to work with a computer using the tangible interface Makey Makey, the approach used did not necessarily result in better results on their tests.

The interaction between the grade and approach factors is statistically significant. Depending on the grade, it is most advisable to use different types of approaches. Only for fifth graders did the use of the blackboard result in a statistically significant improvement over the other types of approaches.

Regarding the grades, approaches and concepts, it can be concluded that any of the approaches can be used for concepts such as loops, conditionals, sequences and outputs, which always show very large effects. For the memory concept, both a blackboard and a PC & mouse have very large effects. However, for the inputs concept, a very large effect is obtained only when using the blackboard. Moreover, the concepts of loops, conditionals, sequences and memory resulted in very large effects in all grades. The outputs concept only results in a large effect for 5th graders. The inputs concept only results in a very large effect for 6th graders.

Future work will include studying the motivations or emotions of working with tangible interfaces such as Makey Makey. Other areas of research include the use of additional factors, such as gender, type of school, or student and teacher motivation with regard to adopting one type of approach or another. Additionally, the use of different times slots or different interaction types for adjusting to new programs or the use of tangible interfaces that can emotionally affect student concentration will be considered. Moreover, studying the effects of additional programming concepts, testing students enrolled in more grades in primary education, and covering higher levels of Bloom's taxonomy will be examined to determine the relationships between all these factors.

ACKNOWLEDGMENT

The authors would also like to thank the school, teachers and students for their collaboration.

REFERENCES

[1] W. W. F. Lau and A. H. K. Yuen, "Modelling programming performance: Beyond the influence of learner characteristics," *Comput. Edu.*, vol. 57, no. 1, pp. 1202–1213, Aug. 2011.
[2] L. Manovich, *Software Takes Command A&C Black*. New York, NY, USA: Academic, 2013.
[3] D. Rushkoff, *Program or be Programmed: Ten Commands for a Digital Age*. New York, NY, USA: Or Books, 2010.
[4] *Education Act*, Government Ireland, Statute Educ. Act, Dublin, Ireland, 1998.
[5] K. Brennan and M. Resnick, *New Frameworks for Studying and Assessing the Development of Computational Thinking*, vol. 1. Vancouver, BC, Canada: American Educational Research Association, 2012, p. 25.

[6] S. Campe and J. Denner, "Programming games for learning: A research synthesis," presented at the Paper Annual Meeting Amer. Educ. Res. Assoc. (AERA), 2015.
[7] M. Jovanov, E. Stankov, M. Mihova, S. Ristov, and M. Gusev, "Computing as a new compulsory subject in the macedonian primary schools curriculum," in *Proc. IEEE Global Eng. Edu. Conf. (EDUCON)*, Apr. 2016, pp. 680–685.
[8] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine, "Learning basic programming concepts by creating games with scratch programming environment," *Procedia Social Behav. Sci.*, vol. 191, pp. 1479–1482, Jun. 2015.
[9] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[10] M. Resnick, F. Martin, R. Sargent, and B. Silverman, "Programmable bricks: Toys to think with," *IBM Syst. J.*, vol. 35, no. 3.4, pp. 443–452, 1996.
[11] E. Lahtinen, K. Ala-Mutka, and H.-M. Järvinen, "A study of the difficulties of novice programmers," *ACM SIGCSE Bull.*, vol. 37, no. 3, pp. 14–18, Sep. 2005.
[12] D. Ginat, "On novice loop boundaries and range conceptions," *Comput. Sci. Edu.*, vol. 14, no. 3, pp. 165–181, Sep. 2004.
[13] O. Seppälä, L. Malmi, and A. Korhonen, "Observations on student misconceptions—A case study of the build—Heap algorithm," *Comput. Sci. Edu.*, vol. 16, no. 3, pp. 241–255, Sep. 2006.
[14] L. J. Barker, C. McDowell, and K. Kalahar, "Exploring factors that influence computer science introductory course students to persist in the major," *ACM SIGCSE Bull.*, vol. 41, no. 1, pp. 153–157, Mar. 2009.
[15] N. J. Coull and I. M. M. Duncan, "Emergent requirements for supporting introductory programming," *Innov. Teaching Learn. Inf. Comput. Sci.*, vol. 10, no. 1, pp. 78–85, Feb. 2011.
[16] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and J. T. Korb, "Computational thinking in elementary and secondary teacher education," *ACM Trans. Comput. Edu.*, vol. 14, no. 1, pp. 1–16, Mar. 2014.
[17] A. Yadav, S. Gretter, S. Hambrusch, and P. Sands, "Expanding computer science education in schools: Understanding teacher experiences and challenges," *Comput. Sci. Edu.*, vol. 26, no. 4, pp. 235–254, Dec. 2016.
[18] M. Feldgen and O. Clua, "Games as a motivation for freshman students to learn programming," in *Proc. 34th Annu. Frontiers Edu. FIE*, Oct. 2004, pp. S1H/11–S1H/16.
[19] T. Jenkins, "The motivation of students of programming," *ACM SIGCSE Bull.*, vol. 33, no. 3, pp. 53–56, Sep. 2001.
[20] M. Huggard and C. M. Goldrick, "Practical positioning projects: Location based services in the laboratory," in *Proc. Frontiers Edu. 35th Annu. Conf.*, 2005, pp. 1–6.
[21] O. Atrachan, "Hooks and props in teaching programming," in *Proc. ITiCSE*, Dublin, Ireland, 1998, pp. 21–24.
[22] E. V. Siegel, "Why do fools fall into infinite loops: Singing to your computer science class," in *Proc. 4th Annu. SIGCSE/SIGCUE ITiCSE Conf. Innov. Technol. Comput. Sci. Edu. - ITiCSE*, 1999, pp. 167–170.
[23] S. Cooper, W. Dann, and R. Pausch, "Teaching objects-first in introductory computer science," in *Proc. 34th SIGCSE Tech. Symp. Comput. Sci. Edu. - SIGCSE*, 2003, pp. 191–195.
[24] W. Dann, S. Cooper, and R. Pausch, *Learning to Program With Alice*. Upper Saddle River, NJ, USA: Prentice-Hall, 2006.
[25] G.-A. Amoussou and S. Steinberg, "Work in progress: Assessing the teaching of the science of design in computer science programs," in *Proc. Frontiers Educ., 36th Annu. Conf.*, Oct. 2006, pp. 27–28.
[26] S. J. Lincke, "Work in progress—Motivating students for software engineering," in *Proc. Frontiers Edu. 35th Annu. Conf.*, Oct. 2005, pp. 1–2.
[27] A. F. McKenna, J. Nocedal, R. Freeman, and S. H. Carr, "Introducing a constructivist approach to applying programming skills in engineering analysis," in *Proc. Frontiers Edu. 35th Annu. Conf.*, Oct. 2005, Art. no. 1611958.
[28] C. A. Wellington, T. Briggs, and C. D. Girard, "Comparison of student experiences with plan-driven and agile methodologies," in *Proc. Frontiers Edu. 35th Annu. Conf.*, Oct. 2005, Art. no. T3G-18.
[29] S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas*. New York, NY, USA: Basic Books, 1980.
[30] P. Blikstein, "Digital fabrication and 'making' in education: The democratization of invention," *FabLabs, Mach., Makers and Inventors*, vol. 4, no. 1, pp. 1–21, 2013.
[31] L. Martin, "The promise of the maker movement for education," *J. Pre-College Eng. Edu. Res. (J-PEER)*, vol. 5, no. 1, p. 4, Apr. 2015.
[32] S. Papert, *The Children's Machine*. New York, NY, USA: Basic Books, 1993.
[33] F. Heintz, L. Mannila, and T. Farnqvist, "A review of models for introducing computational thinking, computer science and computing in K-12 education," in *Proc. IEEE Frontiers Edu. Conf. (FIE)*, Oct. 2016, pp. 1–9.
[34] Q. Burke and Y. B. Kafai, "The writers' workshop for youth programmers: Digital storytelling with scratch in middle school classrooms," in *Proc. 43rd ACM Tech. Symp. Comput. Sci. Edu. - SIGCSE*, 2012, pp. 433–438.
[35] D. Franklin, G. Skifstad, R. Rolock, I. Mehrotra, V. Ding, A. Hansen, D. Weintrop, and D. Harlow, "Using upper-elementary student performance to understand conceptual sequencing in a blocks-based curriculum," in *Proc. ACM SIGCSE Tech. Symp. Comput. Sci. Edu.*, Mar. 2017, pp. 231–236, doi: 10.1145/3017680.3017760.
[36] M. Horn and M. Bers. "Tangible Computing," in *The Cambridge Handbook of Computing Education Research*, S. A. Fincher and A. V. Robins, Eds. Cambridge, U.K.: Cambridge Univ. Press, 2019.
[37] B. M. Collective and D. Shaw, "Makey makey: Improvising tangible and nature-based user interfaces," in *Proc. 6th Int. Conf. Tangible, Embedded Embodied Interact.*, Feb. 2012, pp. 367–370.
[38] E. Lee, Y. B. Kafai, V. Vasudevan, and R. L. Davis, "Playing in the arcade: Designing tangible interfaces with MaKey MaKey for Scratch games," in *Playful User Interfaces*. Singapore: Springer, 2014, pp. 277–292.
[39] S. Cuendet, J. Dehler-Zufferey, G. Ortoleva, and P. Dillenbourg, "An integrated way of using a tangible user interface in a classroom," *Int. J. Comput.-Supported Collaborative Learn.*, vol. 10, no. 2, pp. 183–208, Jun. 2015.
[40] Y. Zhou, "Tangible user interfaces in learning and education," *Social Behav. Sci.*, vol. 1, pp. 20–25, Jan. 2015.
[41] B. Schneider and P. Blikstein, "Comparing the benefits of a tangible user interface and contrasting cases as a preparation for future learning," in *Proc. Exploring Mater. Conditions Learn., Comput. Supported Collaborative Learn. (CSCL) Conf.*, vol. 1, O. Lindwall, P. Häkkinen, T. Koschman, P. Tchounikine, and S. Ludvigsen, Eds. Gothenburg, Sweden: International Society of the Learning Sciences (ISLS), 2015.
[42] G. E. Baykal, I. V. Alaca, A. E. Yantaç, and T. Göksun, "A review on complementary natures of tangible user interfaces (TUIs) and early spatial learning," *Int. J. Child-Comput. Interact.*, vol. 16, pp. 104–113, Jun. 2018.
[43] A. Skulmowski, S. Pradel, T. Kühnert, G. Brunnett, and G. D. Rey, "Embodied learning using a tangible user interface: The effects of haptic perception and selective pointing on a spatial learning task," *Comput. Edu.*, vols. 92–93, pp. 64–75, Jan. 2016.
[44] M. S. Horn, R. J. Crouser, and M. U. Bers, "Tangible interaction and learning: The case for a hybrid approach," *Pers. Ubiquitous. Comput.*, vol. 16, no. 4, pp. 379–389, 2012, doi: 10.1007/s00779-011-0404-2.
[45] C.-Y. Lin and Y.-M. Chang, "Increase in physical activities in kindergarten children with cerebral palsy by employing MaKey–MaKey-based task systems," *Res. Develop. Disabilities*, vol. 35, no. 9, pp. 1963–1969, Sep. 2014.
[46] Y. B. Kafai and V. Vasudevan, "Constructionist gaming beyond the screen: Middle school Students' crafting and computing of touchpads, board games, and controllers," in *Proc. Workshop Primary Secondary Comput. Edu. ZZZ - WiPSCE*, 2015, pp. 49–54.
[47] E. Sun and S. Han, "Fun with bananas: Novel inputs on enjoyment and task performance," presented at the CHI Extended Abstr. Hum. Factors Comput. Syst., 2013, pp. 1275–1280.
[48] F. J. García-Peñalvo, D. Reimann, M. Tuul, A. Rees, and I. Jormanainen, "An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers," Erasmus + KA2 project 'TACCLE 3 – Coding', Tech. Rep. 2015-1-BE02-KA201-012307, 2016.
[49] S.-C. Kong, M. M. Chiu, and M. Lai, "A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education," *Comput. Edu.*, vol. 127, pp. 178–189, Dec. 2018.
[50] G. Chen, J. Shen, L. Barth-Cohen, S. Jiang, X. Huang, and M. Eltoukhy, "Assessing elementary students' computational thinking in everyday reasoning and robotics programming," *Comput. Edu.*, vol. 109, pp. 162–175, Jun. 2017.
[51] A. Sovic, T. Jagust, and D. Sersic, "How to teach basic university-level programming concepts to first graders?" in *Proc. IEEE Integr. STEM Edu. Conf.*, Mar. 2014, pp. 1–6.
[52] F. Kalelioğlu, "A new way of teaching programming skills to K-12 students: Code.org," *Comput. Hum. Behav.*, vol. 52, pp. 200–210, Nov. 2015.

[53] C. Brady, K. Orton, D. Weintrop, G. Anton, S. Rodriguez, and U. Wilensky, "All roads lead to computing: Making, participatory simulations, and social computing as pathways to computer science," *IEEE Trans. Educ.*, vol. 60, no. 1, pp. 59–66, Feb. 2017.
[54] J. Shim, D. Kwon, and W. Lee, "The effects of a robot game environment on computer programming education for elementary school students," *IEEE Trans. Educ.*, vol. 60, no. 2, pp. 164–172, May 2017.
[55] S. Popat and L. Starkey, "Learning to code or coding to learn? A systematic review," *Comput. Edu.*, vol. 128, pp. 356–376, 2019.
[56] M. Johnson and G. Lakoff, *Metaphors we Live by*. Chicago, IL, USA: Univ. Chicago Press., 2003.
[57] M. C. Bateson, *Peripheral Visions: Learning Along the Way*. New York, NY, USA: HarperCollins, 1994.
[58] C. J. Craig, "Metaphors of knowing, doing and being: Capturing experience in teaching and teacher education," *Teaching Teacher Edu.*, vol. 69, pp. 300–311, Jan. 2018.
[59] T. H. Hui and I. N. Umar, "Does a combination of metaphor and pairing activity help programming performance of students with different self-regulated learning level," *TOJET, Turkish Online J. Educ. Technol.*, vol. 10, no. 4, pp. 121–129, 2011.
[60] M. Forišek and M. Steinová, "Metaphors and analogies for teaching algorithms," presented at the 43rd ACM Tech. Symp. Comput. Sci. Educ. 2012.
[61] D. Gentner and A. B. Markman, "Structure mapping in analogy and similarity," *Amer. Psychologist*, vol. 52, no. 1, p. 45, 1997.
[62] B. Falkenhainer, K. D. Forbus, and D. Gentner, "The structure-mapping engine: Algorithm and examples," *Artif. Intell.*, vol. 41, no. 1, pp. 1–63, Nov. 1989.
[63] S. Schez-Sobrino, M. Á. García, C. Gómez, D. Vallejo, A. I. Molina, C. Lacave, C. Glez-Morcillo, J. A. Albusac, and M. Á. Redondo, "ANGELA: A novel approach of graphic notation based on the metaphor of road signs to facilitate the learning of programming," in *Proc. 7th Int. Conf. Technol. Ecosyst. Enhancing Multiculturality*, Oct. 2019, pp. 822–829.
[64] J. Hromkovič, T. Kohn, D. Komm, and G. Serafini, "Combining the power of python with the simplicity of logo for a sustainable computer science education," in *Proc. Int. Conf. Informat. Schools, Situation, Evol., Perspect.* Berlin, Germany: Springer, 2016, pp. 155–166.
[65] S. Tarkan, V. Sazawal, A. Druin, E. Golub, E. M. Bonsignore, G. Walsh, and Z. Atrash, "Toque: Designing a cooking-based programming language for and with children," presented at the Comput. Hum.-Interact. Conf., Apr. 2010.
[66] T. Moape, S. Ojo, and E. Van Wyk, "A metaphor-driven interactive multimedia simulation for teaching and learning of programming concepts," presented at the IST-Africa Conf., May 2018.
[67] D. Perez-Marin, R. Hijon-Neira, and M. Martin-Lope, "A methodology proposal based on metaphors to teach programming to children," *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, vol. 13, no. 1, pp. 46–53, Feb. 2018.
[68] D. Pérez-Marín, R. Hijón-Neira, A. Bacelo, and C. Pizarro, "Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children," *Comput. Human Behav.*, vol. 105, pp. 1–10, Apr. 2020, doi: 10.1016/j.chb.2018.12.027.
[69] J. Sorva, V. Karavirta, and L. Malmi, "A review of generic program visualization systems for introductory programming education," *ACM Trans. Comput. Edu.*, vol. 13, no. 4, pp. 1–64, Nov. 2013, doi: 10.1145/2490822.
[70] P. Brusilovsky and T. D. Loboda, "WADEIn II: A case for adaptive explanatory visualization," *ACM SIGCSE Bull.*, vol. 38, no. 3, pp. 48–52, Sep. 2006.
[71] A. N. Kumar, "Results from the evaluation of the effectiveness of an online tutor on expression evaluation," *SIGCSE Bull.*, vol. 37, no. 1, pp. 216–220, 2005.
[72] M. Kölling, "Using BlueJ to introduce programming," in *Reflections on the Teaching of Programming: Methods and Implementations*, J. Bennedsen, M. E. Caspersen, and M. Kolling Eds. Berlin, Germany: Springer, 2008, pp. 98–115.
[73] M. Kölling, "The greenfoot programming environment," *ACM Trans. Comput. Edu.*, vol. 10, no. 4, pp. 1–21, Nov. 2010, doi: 10.1145/1868358.1868361.
[74] J. A. Velázquez-Iturbide, A. Pérez-Carrasco, and J. Urquiza-Fuentes, "SRec: An animation system of recursion for algorithm courses," *SIGCSE Bull.*, vol. 40, no. 3, pp. 225–229, 2008.
[75] L. Ma, J. Ferguson, M. Roper, I. Ross, and M. Wood, "Improving the mental models held by novice programmers using cognitive conflict and jeliot visualisations," *ACM SIGCSE Bull.*, vol. 41, no. 3, pp. 166–170, Aug. 2009.
[76] T.-C. Hsu, S.-C. Chang, and Y.-T. Hung, "How to learn and how to teach computational thinking: Suggestions based on a review of the literature," *Comput. Edu.*, vol. 126, pp. 296–310, Nov. 2018.
[77] A. Jedlitschka and D. Pfahl, "Reporting guidelines for controlled experiments in software engineering," presented at the IEEE Int. Symp. Empirical Softw. Eng., Nov. 2005.
[78] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Berlin, Germany: Springer, 2012.
[79] Y. Rogers, J. Paay, M. Brereton, K. L. Vaisutis, G. Marsden, and F. Vetere, "Never too old: Engaging retired people inventing the future with MaKey MaKey," in *Proc. 32nd Annu. ACM Conf. Hum. Factors Comput. Syst. CHI*, 2014, pp. 3913–3922.
[80] T.-C. Yang, S. Y. Chen, and G.-J. Hwang, "The influences of a two-tier test strategy on student learning: A lag sequential analysis approach," *Comput. Edu.*, vol. 82, pp. 366–377, Mar. 2015.
[81] B. S. Bloom, "Mastery learning," in *Mastery Learning: Theory and Practice*, J. H. Block Ed. New York, NY, USA: Holt, Rinehart & Winston, 1971.
[82] D. Davis and J. Sorrell. (1995). *Mastery Learning in Public Schools*. Educational Psychology, Valdosta State University. Accessed: Aug. 5, 2010. [Online]. Available: http://teach.valdosta.edu/whuitt/files/mastlear.html
[83] J. M. Sáez-López, M. Román-González, and E. Vázquez-Cano, "Visual programming languages integrated across the curriculum in elementary school: A two year case study using, 'Scratch' five schools," *Comput. Edu.*, vol. 97, pp. 129–141, Jun. 2016.
[84] D. R. Krathwohl, "A revision of Bloom's taxonomy: An overview," *Theory Into Pract.*, vol. 41, no. 4, pp. 212–218, Nov. 2002.
[85] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Evanston, IL, USA: Routledge, 2013.
[86] C. Brackmann, D. Barone, A. Casali, R. Boucinha, and S. Munoz-Hernandez, "Computational thinking: Panorama of the americas," in *Proc. Int. Symp. Comput. Edu. (SIIE)*, Sep. 2016, pp. 1–6.
[87] R. A. Alamer, W. A. Al-Doweesh, H. S. Al-Khalifa, and M. S. Al-Razgan, "Programming unplugged: Bridging CS unplugged activities gap for learning key programming concepts," in *Proc. 5th Int. Conf. e-Learn. (econf)*, Oct. 2015, pp. 97–103.
[88] T. D. Cook and D. T. Campbell, *Quasi-Experimentation: Design and Analysis Issues for Field Settings*. Boston, MA, USA: Houghton Mifflin, 1979.
[89] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Boston, MA, USA: Houghton Mifflin, 2002.
[90] G. H. Bracht and G. V. Glass, "The external validity of experiments," *Amer. Educ. Res. J.*, vol. 5, no. 4, pp. 437–474, 1968.
[91] S. F. Davis and R. A. Smith, *An introduction to Statistics and Research Methods: Becoming a Psychological Detective*. London, U.K.: Pearson, 2005.
[92] M. A. Saint-Germain. *Research Methods*. Accessed: Jul. 27, 2020. [Online]. Available: https://web.csulb.edu/~msaintg/ppa696/696exper.htm

RAQUEL HIJÓN-NEIRA received the European Ph.D. degree in computer science, in 2010.

She worked as a Computer Science Engineer for a period of five years and as an Instructor with the university for a period of 20 years. She has been a member of the Laboratory of Information Technologies in Education (LITE) since its inception. She is currently an Assistant Professor with the Computer Science Department, Universidad Rey Juan Carlos, Madrid, Spain. Her research interests include software for and innovation in programming education, educative technologies, teaching programming to K-12 students, and serious games.

Dr. Hijón-Neira received the Best Thesis Award from the Spanish Chapter of the IEEE Education Society.

DIANA PÉREZ-MARIN received the European Ph.D. degree in computer science and telecommunication, in 2007.

She worked as a Lecturer and a Researcher with the Universidad Autónoma de Madrid for a period of ten years. She has been a Lecturer and a Researcher with the Universidad Rey Juan Carlos for a period of ten years. She is currently a Professor with the Computer Science Department, Universidad Rey Juan Carlos, Madrid, Spain. She is a member of the Laboratory of Information Technologies in Education (LITE). She has published more than 100 papers in national and international journals and conferences. Her research interests include human–computer interaction, computer assisted education, and computer science education. She was a recipient of several awards.

CELESTE PIZARRO received the M.Sc. degrees in mathematics and statistics from Universidad de Extremadura, Badajoz, Spain, in 2000 and 2001, respectively, and the Ph.D. degree in computer science and mathematical modeling from the Universidad Rey Juan Carlos, Madrid, Spain, in 2006. She is currently an Assistant Professor in applied mathematics with the Universidad Rey Juan Carlos. She is also an Assistant Professor with the Department of Statistics and Operations Research, Universidad Rey Juan Carlos. Her research interests include different mathematical programming fields (stochastic, integer, and linear programming) and their applications.

CORNELIA CONNOLLY received the B.Eng. degree (Hons.) in computer engineering and the M.Eng. degree (Hons.) for research, and the Ph.D. degree from the University of Limerick (UL), in 2007.

She was a Lecturer in computer and mathematics. She is currently a Lecturer with the School of Education, National University of Ireland Galway, Galway, Ireland. She is also a STEM Teacher/Educator. Her research interests include mathematics and computer science education. She publishes regularly on educational design and pedagogical enhancement in STEM education.

Dr. Connolly was a recipient of the IEEE Student Paper Award at the 2001 IEEE Conference on the History of Telecommunications.